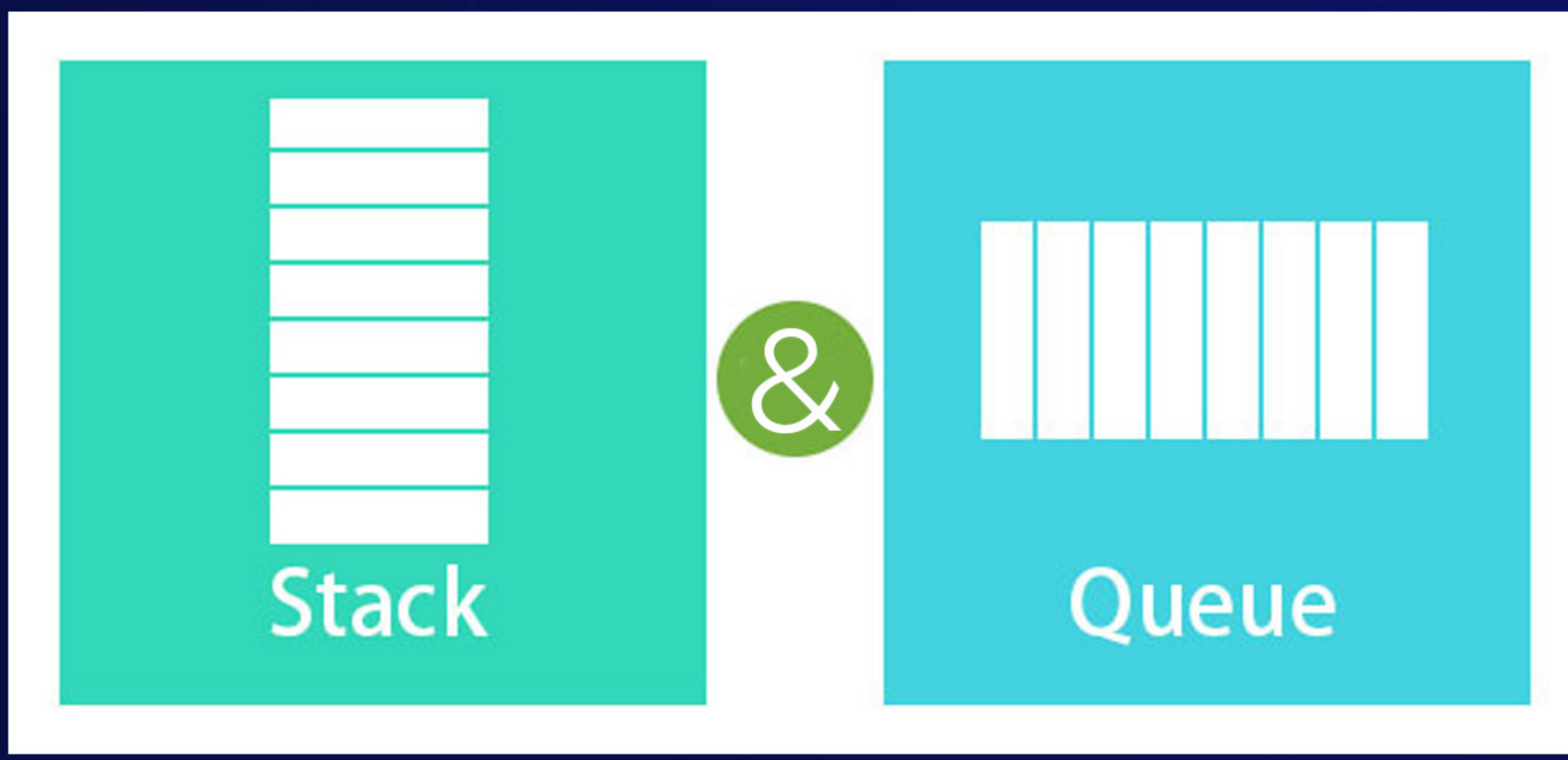
AlgoTutor

MASTER STACK & QUEUES IN JUST 05 DAYS



AlgoTutor

Day 1

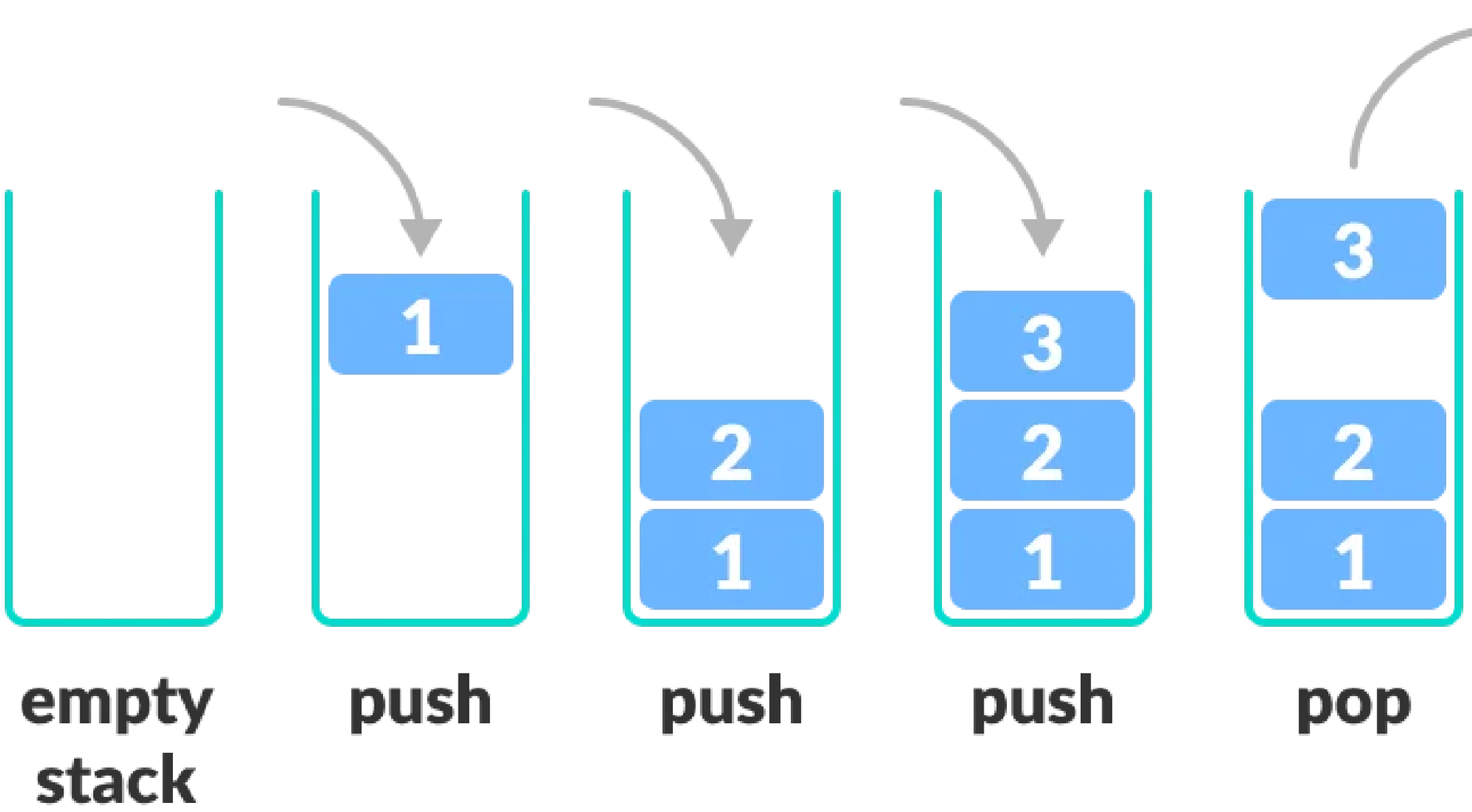
Understanding Stacks

- Learn the basics of stacks, including how they work and their common operations (push, pop, peek).
- Study the Last-In-First-Out (LIFO) property of stacks.

Day 2

Stack Implementation and Applications

- Implement a stack data structure from scratch.
- Explore real-world applications of stacks, such as evaluating expressions and function call management.

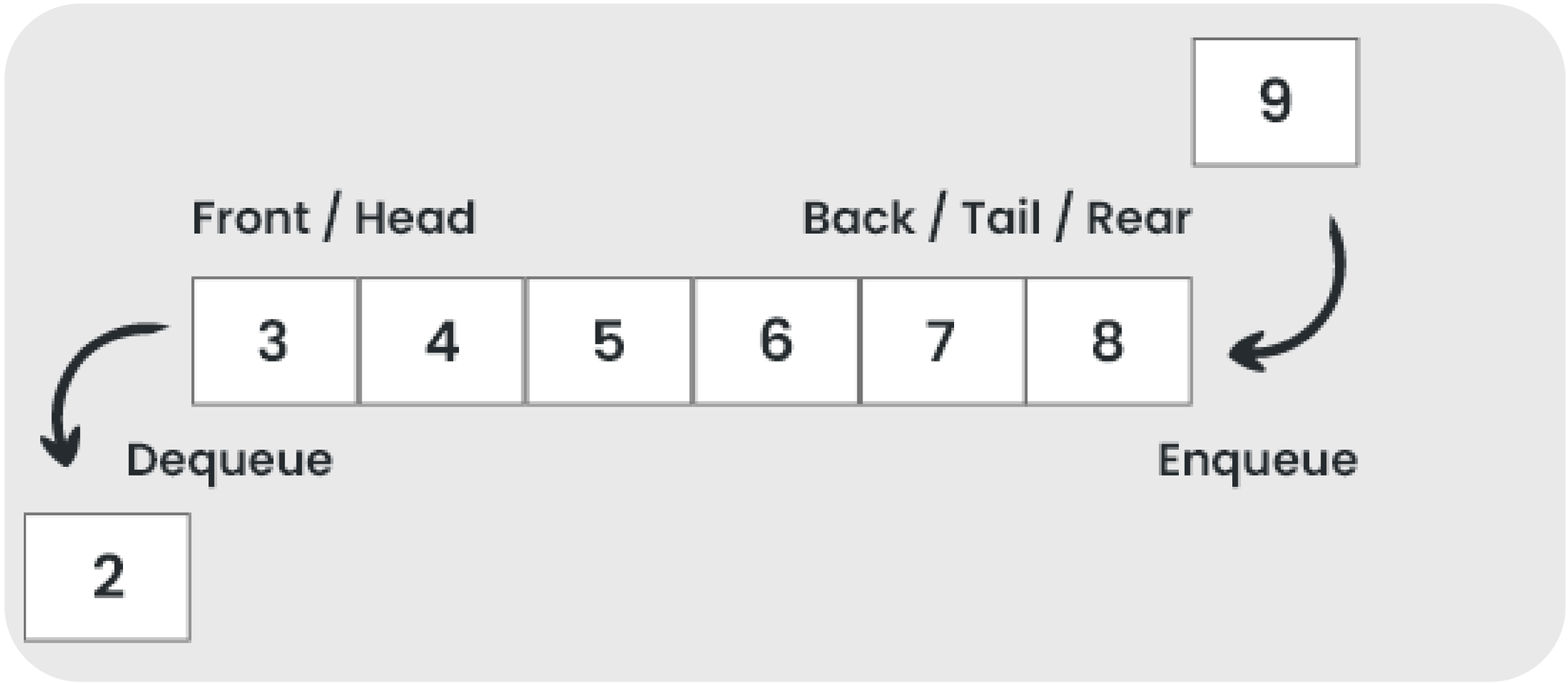


AlgoTutor

Day 3

Queue Basics

- Learn the basics of queues, including their structure and common operations (enqueue, dequeue, front, rear).
- Understand the First-In-First-Out (FIFO) property of queues.



Day 4

Queue Implementation and Applications

- Implement a queue data structure from scratch.
- Study real-world applications of queues, like managing tasks and scheduling.

AlgoTutor

Day 5

Advanced Problems and Review

- Challenge yourself with advanced problems involving both stacks and queues.
- Review the concepts and problem-solving techniques you've learned.

AlgoTutor

Important Practice Questions

01. Implement Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).
Implement the MyStack class:

- void push(int x) Pushes element x to the top of the stack.
- int pop() Removes the element on the top of the stack and returns it.
- int top() Returns the element on the top of the stack.
- boolean empty() Returns true if the stack is empty, false otherwise.

Example 1:

Input

["MyStack", "push", "push", "top", "pop", "empty"]

[[], [1], [2], [], [], []]

Output

[null, null, null, 2, 2, false]

Practice

AlgoTutor

02. Implement Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty).

Implement the MyQueue class:

- void push(int x) Pushes element x to the back of the queue.
- int pop() Removes the element from the front of the queue and returns it.
- int peek() Returns the element at the front of the queue.
- boolean empty() Returns true if the queue is empty, false otherwise.

Example 1:

Input: ["MyQueue", "push", "push", "peek", "pop", "empty"]

[[], [1], [2], [], [], []]

Output: [null, null, null, 1, 1, false]

Practice

AlgoTutor

03. Valid Parentheses

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Example 1:

Input: s = "()"

Output: true

Example 2:

Input: s = "()[]{}"

Output: true

Example 3:

Input: s = "(]"

Output: false

Practice

04. Min Stack

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

- MinStack() initializes the stack object.
- void push(int val) pushes the element val onto the stack.
- void pop() removes the element on the top of the stack.
- int top() gets the top element of the stack.
- int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

Example 1:

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

Output

[null,null,null,null,-3,null,0,-2]

Practice

05. Next Greater Element I

The next greater element of some element x in an array is the first greater element that is to the right of x in the same array.

You are given two distinct 0-indexed integer arrays nums1 and nums2, where nums1 is a subset of nums2.

For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the next greater element of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.

Return an array ans of length nums1.length such that ans[i] is the next greater element as described above.

Example 1:

Input: nums1 = [4,1,2], nums2 = [1,3,4,2]

Output: [-1,3,-1]

Practice

06. Next Greater Element II

Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return the next greater number for every element in nums.

The next greater number of a number x is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

Example 1:

Input: nums = [1,2,1]

Output: [2,-1,2]

Example 2:

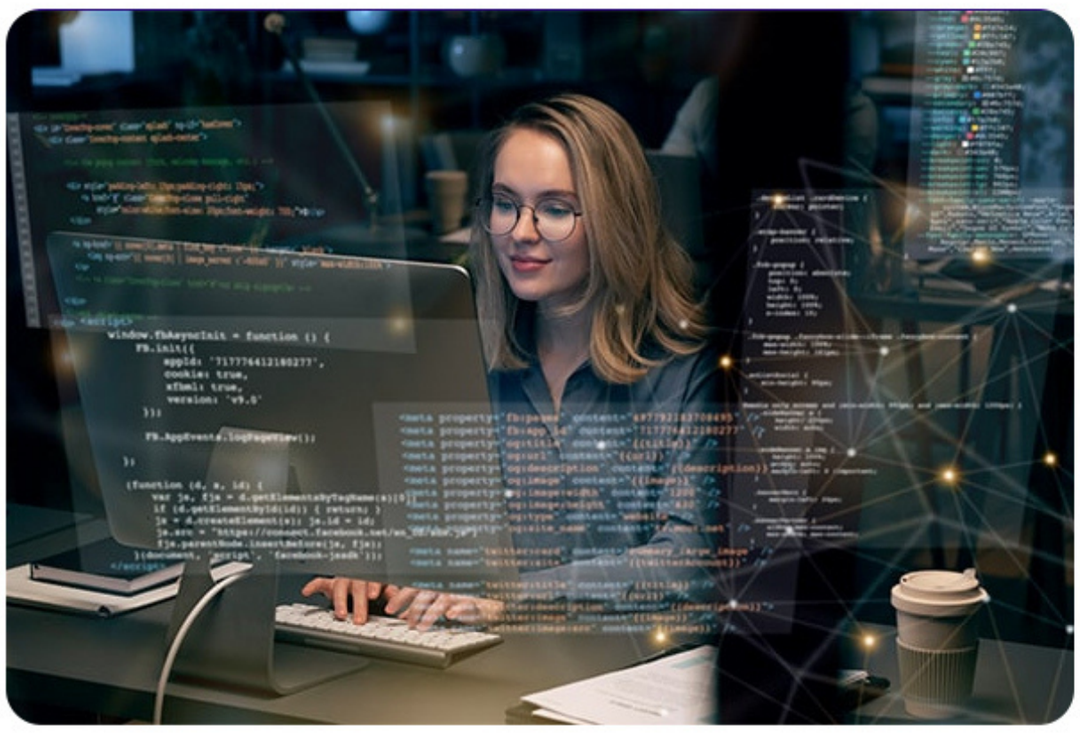
Input: nums = [1,2,3,4,3]

Output: [2,3,4,-1,4]

Practice

AlgoTutor

Take The First Step Towards Fulfilling a Career



Mastering DSA & System Design

Advance Data Science & ML

Full Stack Web Development - MERN

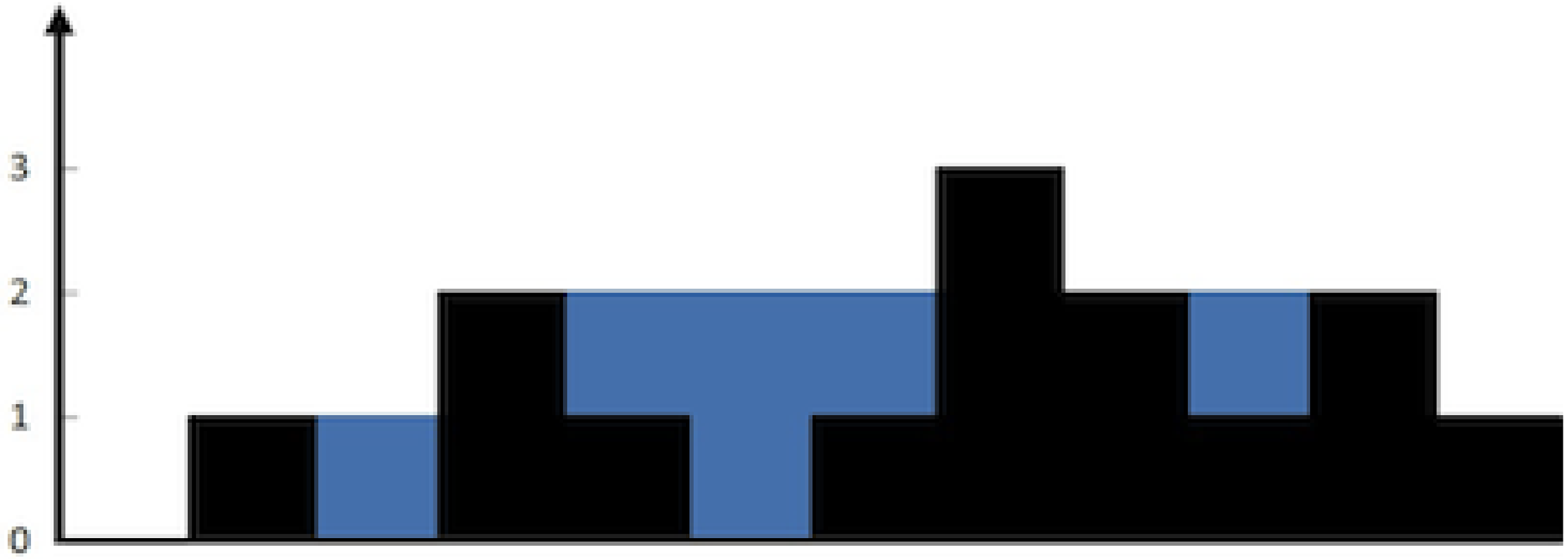
Mastering DSA & System Design with Full Stack Specialization

For Admission Enquiry +91 - 72600 58093

07. Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:



Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Example 2:

Input: height = [4,2,0,3,2,5]

Output: 9

Practice

08. Sum of Subarray Minimums

Given an array of integers arr, find the sum of min(b), where b ranges over every (contiguous) subarray of arr. Since the answer may be large, return the answer modulo 109 + 7.

Example 1:

Input: arr = [3,1,2,4]

Output: 17

Explanation:

Subarrays are [3], [1], [2], [4], [3,1], [1,2], [2,4], [3,1,2], [1,2,4], [3,1,2,4].

Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.

Sum is 17.

Example 2:

Input: arr = [11,81,94,43,3]

Output: 444

Practice

09. Asteroid Collision

We are given an array asteroids of integers representing asteroids in a row.

For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.

Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:

Input: asteroids = [5,10,-5]

Output: [5,10]

Example 2:

Input: asteroids = [8,-8]

Output: []

Practice

10. Sum of Subarray Ranges

You are given an integer array nums. The range of a subarray of nums is the difference between the largest and smallest element in the subarray.

Return the sum of all subarray ranges of nums.

A subarray is a contiguous non-empty sequence of elements within an array.

Example 1:

Input: nums = [1,2,3]

Output: 4

Practice

11. Remove K Digits

Given string num representing a non-negative integer num, and an integer k, return the smallest possible integer after removing k digits from num.

Example 1:

Input: num = "1432219", k = 3

Output: "1219"

Explanation: Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.

Example 2:

Input: num = "10200", k = 1

Output: "200"

Example 3:

Input: num = "10", k = 2

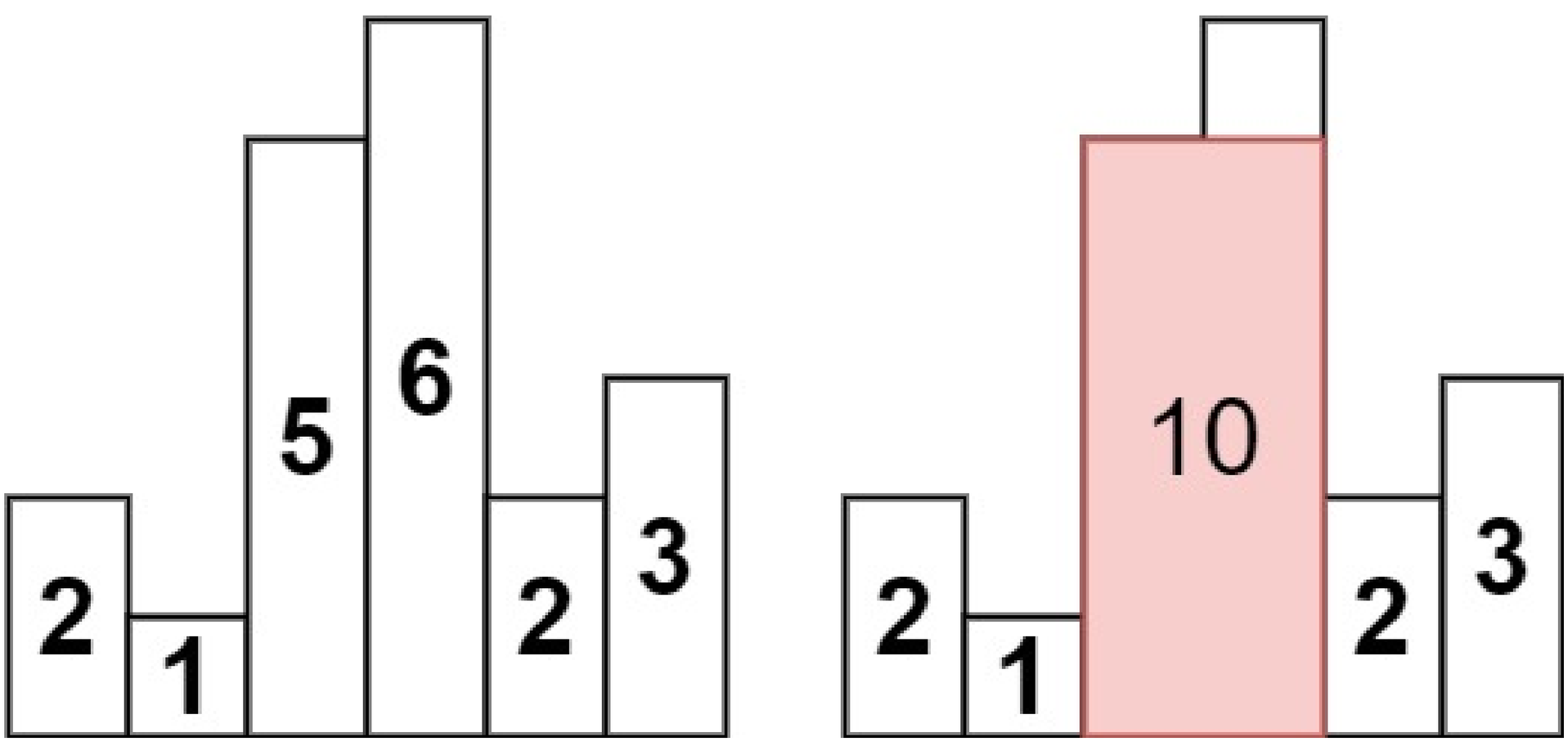
Output: "0"

Practice

12. Largest Rectangle in Histogram

Given an array of integers heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.

Example 1:



Input: heights = [2,1,5,6,2,3]

Output: 10

Explanation: The above is a histogram where width of each bar is 1.

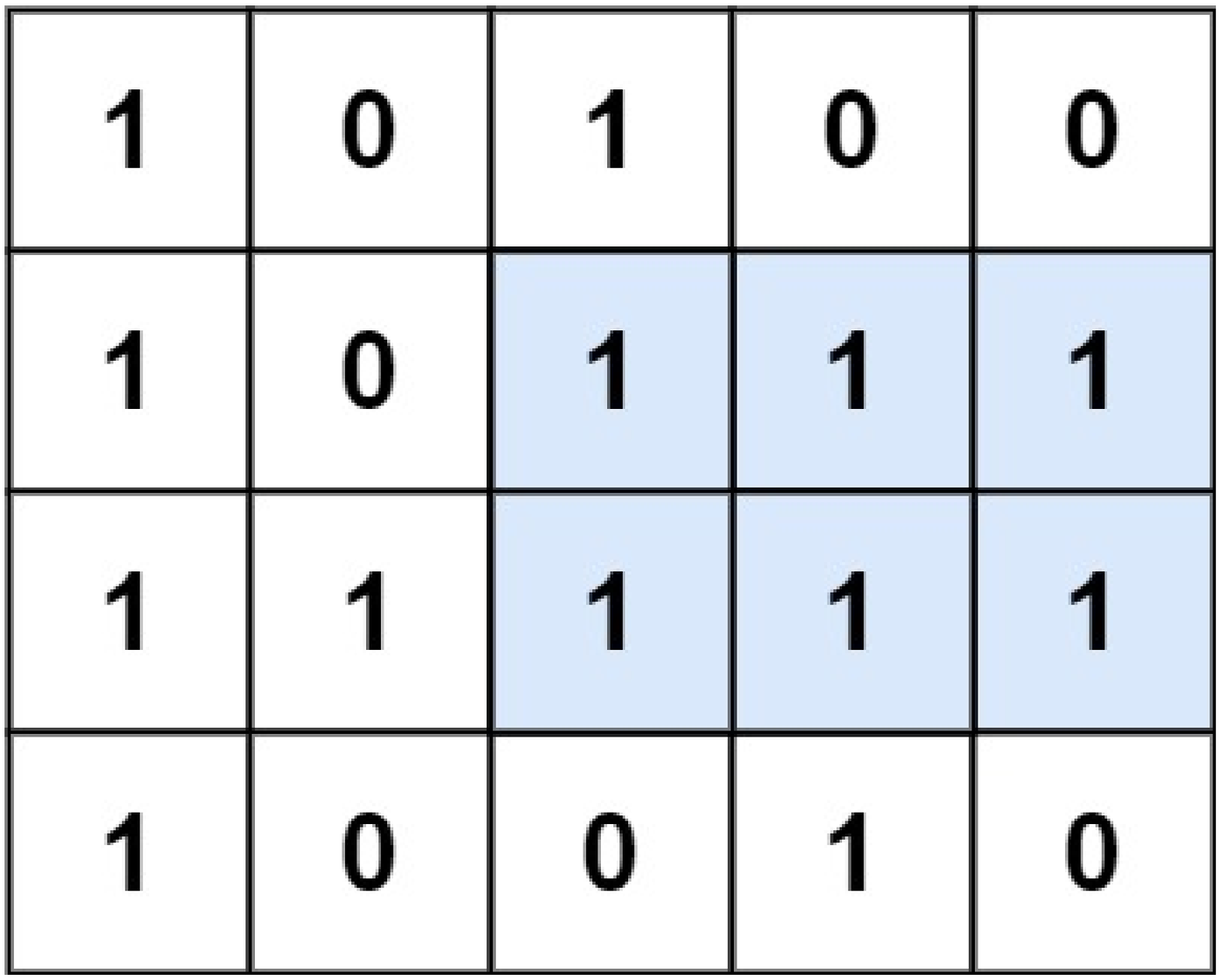
The largest rectangle is shown in the red area, which has an area = 10 units.

Practice

13. Maximal Rectangle

Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

Example 1:



Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]

Output: 6

Explanation: The maximal rectangle is shown in the above picture.

Practice

14. Sliding Window Maximum

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

Example 1:

Input: nums = [1,3,-1,-3,5,3,6,7], k = 3

Output: [3,3,5,5,6,7]

Explanation:

Window position	Max
[1 3 -1] -3 5 3 6 7	3
1 [3 -1 -3] 5 3 6 7	3
1 3 [-1 -3 5] 3 6 7	5
1 3 -1 [-3 5 3] 6 7	5
1 3 -1 -3 [5 3 6] 7	6
1 3 -1 -3 5 [3 6 7]	7

Practice

15. Online Stock Span

Design an algorithm that collects daily price quotes for some stock and returns the span of that stock's price for the current day.

The span of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

- For example, if the prices of the stock in the last four days is [7,2,1,2] and the price of the stock today is 2, then the span of today is 4 because starting from today, the price of the stock was less than or equal 2 for 4 consecutive days.

Example 1:

Input

["StockSpanner", "next", "next", "next", "next", "next", "next", "next", "next"]

[[], [100], [80], [60], [70], [60], [75], [85]]

Output

[null, 1, 1, 1, 2, 1, 4, 6]

Practice

16. LRU Cache

Design a data structure that follows the constraints of a Least Recently Used (LRU) cache.

Implement the LRUCache class:

- LRUCache(int capacity) Initialize the LRU cache with positive size capacity.
- int get(int key) Return the value of the key if the key exists, otherwise return -1.
- void put(int key, int value) Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key.

The functions get and put must each run in O(1) average time complexity.

Example 1:

Input

["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]

[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]

Output

[null, null, null, 1, null, -1, null, -1, 3, 4]

Practice

17. LFU Cache

Design and implement a data structure for a Least Frequently Used (LFU) cache.

Implement the LFUCache class:

- LFUCache(int capacity) Initializes the object with the capacity of the data structure.
- int get(int key) Gets the value of the key if the key exists in the cache. Otherwise, returns -1.
- void put(int key, int value) Update the value of the key if present, or inserts the key if not already present. When the cache reaches its capacity, it should invalidate and remove the least frequently used key before inserting a new item.

Example 1:

Input

["LFUCache", "put", "put", "get", "put", "get", "get", "put", "get", "get", "get"]

[[2], [1, 1], [2, 2], [1], [3, 3], [2], [3], [4, 4], [1], [3], [4]]

Output

[null, null, null, 1, null, -1, 3, null, -1, 3, 4]

Practice

18. Zuma Game

You are playing a variation of the game Zuma.

In this variation of Zuma, there is a single row of colored balls on a board, where each ball can be colored red 'R', yellow 'Y', blue 'B', green 'G', or white 'W'. You also have several colored balls in your hand.

Your goal is to clear all of the balls from the board. On each turn:.

- Pick any ball from your hand and insert it in between two balls in the row or on either end of the row.
- If there is a group of three or more consecutive balls of the same color, remove the group of balls from the board.
    - If this removal causes more groups of three or more of the same color to form, then continue removing each group until there are none left.

Example 1:

Input: board = "WRRBBW", hand = "RB"

Output: -1

Practice

19. Number of Atoms

Given a string formula representing a chemical formula, return the count of each atom.

The atomic element always starts with an uppercase character, then zero or more lowercase letters, representing the name.

One or more digits representing that element's count may follow if the count is greater than 1. If the count is 1, no digits will follow.

- For example, "H2O" and "H2O2" are possible, but "H1O2" is impossible.

Two formulas are concatenated together to produce another formula.

- For example, "H2O2He3Mg4" is also a formula.

A formula placed in parentheses, and a count (optionally added) is also a formula.

- For example, "(H2O2)" and "(H2O2)3" are formulas.

Example 1:

Input: formula = "H2O"

Output: "H2O"

Practice

20. Odd Even Jump

You are given an integer array arr. From some starting index, you can make a series of jumps. The (1st, 3rd, 5th, ...) jumps in the series are called odd-numbered jumps, and the (2nd, 4th, 6th, ...) jumps in the series are called even-numbered jumps. Note that the jumps are numbered, not the indices.

You may jump forward from index i to index j (with i < j) in the following way:

- During odd-numbered jumps (i.e., jumps 1, 3, 5, ...), you jump to the index j such that arr[i] <= arr[j] and arr[j] is the smallest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.
- During even-numbered jumps (i.e., jumps 2, 4, 6, ...), you jump to the index j such that arr[i] >= arr[j] and arr[j] is the largest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.

Example 1:

Input: arr = [10,13,12,14,15]

Output: 2

Practice

AlgoTutor

WHY ALGOTUTOR

- 100% Placement Assistance
- 1-1 personal mentorship from Indusrty experts
- 200+ Successful Alumni
- 147(Avg.)% Salary Hike
- 100% Success Rate
- 23 LPA (Avg.) CTC
- Learn from scratch
- Career Services

For Admission Enquiry

+91-7260058093

info@algotutor.io